#2

500.38828X00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): T. SAKAKIBARA, et al

Serial No.:

Filed: July 28, 2000

Title: COMPUTER SYSTEM

Group:

LETTER CLAIMING RIGHT OF PRIORITY

Honorable Commissioner of
Patents and Trademarks
Washington, D.C. 20231

July 28, 2000

Sir:

Under the provisions of 35 USC 119 and 37 CFR 1.55, the applicant(s) hereby claim(s) the right of priority based on Japanese Patent Application No.(s) 11-216614 filed July 30, 1999.

A certified copy of said Japanese Application is attached.

Respectfully submitted,

ANTONELLI, TERRY, STOUT & KRAUS, LLP

Carl I. Brundidge
Registration No. 29,621

CIB/nac
Attachment
(703) 312-6600

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: １９９９年　７月３０日

出願番号
Application Number: 平成１１年特許願第２１６６１４号

出願人
Applicant (s): 株式会社日立製作所

CERTIFIED COPY OF
PRIORITY DOCUMENT

２０００年　６月２３日

特許庁長官
Commissioner,
Patent Office 近藤隆彦

【書類名】　特許願

【整理番号】　K9052331

【提出日】　平成11年 7月30日

【あて先】　特許庁長官　殿

【国際特許分類】　G06F 12/08

【発明者】

【住所又は居所】　神奈川県秦野市堀山下１番地　株式会社　日立製作所エンタープライズサーバ事業部内

【氏名】　榊原　忠幸

【発明者】

【住所又は居所】　神奈川県秦野市堀山下１番地　株式会社　日立製作所エンタープライズサーバ事業部内

【氏名】　大原　功

【発明者】

【住所又は居所】　東京都国分寺市東恋ケ窪一丁目２８０番地　株式会社日立製作所　中央研究所内

【氏名】　明石　英也

【発明者】

【住所又は居所】　東京都国分寺市東恋ケ窪一丁目２８０番地　株式会社日立製作所　中央研究所内

【氏名】　對馬　雄次

【発明者】

【住所又は居所】　神奈川県横浜市戸塚区吉田町２９２番地　株式会社　日立製作所　生産技術研究所内

【氏名】　村岡　論

【特許出願人】

【識別番号】　000005108

【氏名又は名称】　株式会社　日立製作所

【代表者】　庄山　悦彦

【代理人】

【識別番号】　100073760

【弁理士】

【氏名又は名称】　鈴木　誠

【手数料の表示】

【予納台帳番号】　011800

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　コンピュータシステム

【特許請求の範囲】

【請求項１】　ＣＰＵと、メモリと、前記ＣＰＵと前記メモリの間の階層に位置するキャッシュとを具備するコンピュータシステムにおいて、

前記ＣＰＵからのリクエストが前記キャッシュにヒットするかを判断して、前記キャッシュあるいは前記メモリに対してリクエストを発行するコヒーレントコントローラと、前記コヒーレントコントローラからのリクエストに従って前記キャッシュに登録されているデータの読み書きを制御するキャッシュデータコントローラとを備え、

前記コヒーレントコントローラは、前記ＣＰＵからリードリクエストを受け付けると、前記キャッシュのヒット判定を行うとともに前記キャッシュデータコントローラに投棄リードリクエストを発行し、ヒット判定がキャッシュヒットの場合には前記キャッシュデータコントローラにリードリクエストを発行する手段を有し、

前記キャッシュデータコントローラは、前記コヒーレントコントローラからの投棄リードリクエストを受け付けると、前記キャッシュからデータを読み出して保持する手段と、前記コヒーレントコントローラからリードリクエストを受け付けると、前記保持しておいた投棄リードデータを前記ＣＰＵへのリードリプライデータとして送出する手段を有する、

ことを特徴とするコンピュータシステム。

【請求項２】　請求項１記載のコンピュータシステムにおいて、

前記コヒーレントコントローラは、ヒット判定がキャッシュミスの場合には、キャッシュデータコントローラに投棄リードデータ破棄リクエストを発行する手段を有し、

前記キャッシュデータコントローラは、前記コヒーレントコントローラから投棄リードデータ破棄リクエストを受け付けると、当該投棄リードデータ破棄リクエストに対応する投棄リードリクエストの投棄リードデータを破棄する手段を有する、

ことを特徴とするコンピュータシステム。

【請求項３】　請求項１もしくは２記載のコンピュータシステムにおいて、キャッシュはｎウエイアソシアティブキャッシュからなり、キャッシュデータコントローラは、コヒーレントコントローラからｎウエイ分の投棄リードリクエストを受け付けて、キャッシュからｎウエイ分のデータを読み出して保持することを特徴とするコンピュータシステム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、コンピュータシステムに係り、特にＣＰＵとメインメモリの間にキャッシュを具備するコンピュータシステムにおいて、キャッシュデータの投棄的リード制御方式に関する。

【０００２】

【従来の技術】

近年、ＣＰＵの性能向上はメモリの性能向上に比べて遥かに高くその性能差はますます広がる方向にある。メインメモリより高速でメインメモリの内容の一部を記憶するキャッシュは、このようなＣＰＵとメモリの性能差を吸収し、実効メモリアクセス時間を短縮するために使用される。

【０００３】

コンピュータシステムにおけるキャッシュの容量の選択は、そのコンピュータシステムの構成に大きく依存する。高性能なＣＰＵには、それ自体が大容量なキャッシュを持つ場合が多い。このようなＣＰＵを数多く接続する場合、２～８程度のＣＰＵをバスまたはスイッチで接続し、さらにそれ以上のＣＰＵを接続する場合には、それらを又、バスまたはスイッチで接続するといった多階層のメモリシステムを構成することが多い。このような構成をとると、ＣＰＵとメインメモリの間のアクセスレイテンシが増加し、ＣＰＵでキャッシュミスが生じた場合の性能に大きな影響を及ぼす。このため、最も上の階層の２～８程度のＣＰＵを接続したバスあるいはスイッチ制御の階層にキャッシュを設けて、ＣＰＵのキャッシュあふれが生じた時の性能低下を防ぐ必要がある。例えば、特開平９－１２８

３４６号公報には、このような階層バスシステムのキャッシュ構成例が開示されている。このときのキャッシュ容量は、その上につながる全てのＣＰＵの総キャッシュ容量以上である必要がある。これはもしＣＰＵキャッシュと同じあるいは小さいキャッシュ容量しか持たない場合には、ＣＰＵキャッシュであふれが生じた時に、その下の階層でもキャッシュあふれが生じやすくなり、著しいシステム性能の低下を招く恐れがあるためである。

【０００４】

ところで、キャッシュの高速アクセスを実現するために、キャッシュにＳＲＡＭ等の高速な素子を用いることが一般的である。通常、このＳＲＡＭ内の同一ロケーションにキャッシュタグ（タグアドレス）とキャッシュデータを格納して、リードリクエストを処理する場合には、キャッシュタグとキャッシュデータを同時に読み出し、リクエストアドレスでキャッシュタグをチェックして、ヒットしていれば直ちにキャッシュデータを使うことができるように構成する。しかし、ＳＲＡＭはメインメモリ等に使用されるＤＲＡＭと比較して集積度が一桁以上低く、大容量キャッシュを構成するにはＳＲＡＭを多数用いる必要がある。ＳＲＡＭを多数用いる場合、多数のＳＲＡＭとのインタフェースをとるため、キャッシュを制御するＬＳＩのピン数が増加して一つのＬＳＩで構成できなくなる。キャッシュタグ部分はキャッシュヒットチェックに使用し、このヒットチェックにかかる時間の増加は、直接メモリアクセスレイテンシの増加につながるため、キャッシュタグ部分とのインタフェースを持つＬＳＩはＣＰＵバスと同一ＬＳＩに入れる必要がある。キャッシュデータ部分とのインタフェースを持つＬＳＩはＣＰＵバスを含むＬＳＩと異なるＬＳＩとし、そのインタフェースをＣＰＵバス程度のデータ幅とすることによって、ＬＳＩのピンネックを解消することができる。

【０００５】

一方、キャッシュのヒット率を向上させる方式として、セットアソシアティブ方式がある。例えば、特開平５－２２５０５３号公報にセットアソシアティブキャッシュのタグ比較を行う方式およびその高速化について開示されている。

【０００６】

セットアソシアティブ方式におけるヒットチェックでは、複数ウエイの複数キ

ャッシュラインのキャッシュタグを読み出して、複数ライン同時にヒットチェックを行う。このとき、複数ラインのどのデータを使用するかはキャッシュヒットチェックが完了するまでわからない。ＣＰＵ内部に搭載されているキャッシュ（オンチップキャッシュ）では、キャッシュデータの読み出しをキャッシュタグの読み出しと同時に行い、キャッシュヒットチェックが完了した後で必要とするデータだけを選択することによってキャッシュアクセスレイテンシを削減する方式をとることが一般的である。

【０００７】

図１２に、この種のセットアソシアティブキャッシュの構成例を示す。図１２はＮエントリからなる４ウエイセットアソシアティブキャッシュを示している。各エントリは４つのウエイ、第０ウエイ１０００、第１ウエイ１００１、第２ウエイ１００２、第３ウエイ１００３からなる。キャッシュに含まれる情報は、キャッシュの状態（有効、無効）を示すＳＴＡＴ１００４、キャッシュタグ（アドレス）１００５、キャッシュデータ１００６からなる。一般にキャッシュのエントリ番号はメモリアドレスの下位アドレスを用い、上位アドレスをキャッシュタグとする方法がとられる。オンチップキャッシュでは、図１２の様にキャッシュタグ１００５とキャッシュデータ１００６をあわせて格納してあるため、該当エントリの各ウエイ１０００～１００３のキャッシュタグ１００５とキャッシュデータ１００６を同時に読み出して、キャッシュヒットしたウエイ番号を使って直ちにデータを選択することができる。

【０００８】

しかし、大容量セットアソシアティブキャッシュを実現しようとすると、キャッシュデータとインタフェースを持つＬＳＩと、ＣＰＵバスとキャッシュタグとのインタフェースを含むＬＳＩとを分離せざるをえず、この場合にはキャッシュタグとキャッシュデータを同時に読み出すことはできない。そこで、キャッシュタグとキャッシュデータを別々に読み出すことになるが、このとき、物理的な制約によってキャッシュタグとインタフェースを持つＬＳＩとキャッシュデータとインタフェースを持つＬＳＩとの間のデータ幅がＣＰＵバス程度しかない場合、複数ライン分のキャッシュデータを全てＣＰＵバス側のＬＳＩまで読み出すには

時間がかかり過ぎてしまう。例えば、ＣＰＵバス幅が８バイトで、ラインサイズが３２バイトの場合、４ウエイ分のラインをキャッシュデータ側ＬＳＩからキャッシュタグ側ＬＳＩに転送するのに４サイクル×４ウエイ＝１６サイクルかかってしまう。これはキャッシュを引く度に１６サイクルかかってしまうことを意味しており、著しく性能が低下してしまう。この性能低下を防ぐには、キャッシュヒットチェック結果が分かってからキャッシュデータを読み出す必要が出てくるが、これではキャッシュのアクセスレイテンシの増大を引き起こすことになってしまう。

【０００９】

【発明が解決しようとする課題】

上記のように、セットアソシアティブ方式等の大容量キャッシュをＣＰＵとメインメモリの間に設ける場合、ＬＳＩのピン数などの制約からキャッシュタグ部分とキャッシュデータ部分を別々のＬＳＩに分けて管理する必要性が出てくる。そのような構成をとった場合、キャッシュタグを読み出してキャッシュヒットチェックを行ってからキャッシュデータを読み出すとキャッシュ読み出しレイテンシが増大するという問題があった。

【００１０】

本発明の目的は、ＣＰＵとメインメモリの間の階層に位置するｎウエイセットアソシアティブキャッシュ等のキャッシュを具備するコンピュータシステムにおいて、上記のようにキャッシュタグ部分とキャッシュデータ部分を別ＬＳＩで管理した場合のキャッシュデータの読み出し時間の短縮を実現することにある。

【００１１】

【課題を解決するための手段】

上記目的を達成するため、本発明では、キャッシュヒットチェックを行う前に、キャッシュデータ部分のコントローラに投棄的なリードリクエストを発行して、キャッシュからのデータを先読みして保持しておき、キャッシュヒットチェックが終了した時点で、キャッシュヒットした場合、投棄的にリードしておいたデータを使用することによって、キャッシュヒットチェック時間を短縮してキャッシュデータを読み出すようにしたものである。

【００１２】

【発明の実施の形態】

以下、本発明の一実施の形態について図面により説明する。

図１に、本発明の一実施の形態のコンピュータシステムを示す。本システムは、２つのＣＰＵ（０）１、ＣＰＵ（１）２、ストレージコントローラ（ＳＣＵ）４、キャッシュタグ部５、キャッシュデータコントローラ６、キャッシュデータ部７、メインメモリ８、ＣＰＵ（０）１とＣＰＵ（１）２とＳＣＵ４を接続するバス３などから構成される。さらにＳＣＵ４はバス１６、メモリアクセスリクエストキュー１７、ライトデータバッファ１８、リードリプライデータバッファ１９、コヒーレントコントローラ２０、メモリアクセスコントローラ２１、バス２２と、それらを接続するバス２３～２９から構成される。なお、ここではＣＰＵの数（ノード数）は２つであるが、勿論、それ以上のノード数でも、また、１つでもかまわない。

【００１３】

図示しないが、ＣＰＵ（０）１とＣＰＵ（１）２には内臓キャッシュが設けられているものと仮定する。また、キャッシュタグ部５とキャッシュデータ部７は高速なメモリ素子であるＳＲＡＭなどで構成され、メインメモリ８は低速なメモリ素子であるＤＲＡＭなどで構成されているものとする。さらに、キャッシュタグ部５およびキャッシュデータ部７で構成される本システムのキャッシュは、４ウエイセットアソシアティブキャッシュであるものとする。

【００１４】

図２に、ＣＰＵ（０）１およびＣＰＵ（１）２からのリクエストアドレスとキャッシュタグ、キャッシュエントリ番号の関係を示す。本実施形態では、ＣＰＵ（０）１およびＣＰＵ（１）２からのリクエストアドレスは３２ビットであり、キャッシュエントリ数は２５６Ｋエントリであるとする。また、キャッシュラインサイズは３２バイトであると仮定する。図２において、１００はＣＰＵ（０）１およびＣＰＵ（１）２から出力されるリクエストアドレス（ＡＤＲ＜３１：０＞）を示す。キャッシュラインサイズは３２バイトであるため、ＡＤＲ１００の下位６ビットはキャッシュライン内のアドレスを示すことになる。キャッシュエ

ントリ数は２５６Ｋエントリであるため、ＡＤＲ＜２４：７＞の１８ビットがキャッシュエントリ番号１０２となる。そして、残りの上位アドレスであるＡＤＲ＜３１：２５＞がキャッシュタグ１０１となる。

【００１５】

図３および図４にキャッシュタグ部５とキャッシュデータ部７の構成例を示す。図３に示すように、キャッシュタグ部５は２５６Ｋエントリ、４ウエイのキャッシュ状態（ＳＴＡＴ）２１０とキャッシュタグ２１１の集合体である。また、図４に示すように、キャッシュデータ部７は２５６Ｋエントリ、４ウエイのキャッシュデータ２２０の集合体である。キャッシュタグ部５とキャッシュデータ部７の各エントリ、各ウエイは一対一に対応し、例えば、キャッシュデータ部７の０エントリ、０ウエイのブロックに格納されたキャッシュデータに対応するキャッシュタグは、キャッシュタグ部５の０エントリ、０ウエイのブロックに格納される。キャッシュ状態（ＳＴＡＴ）２１０は、当該ブロックのキャッシュデータ（キャッシュライン）の有効、無効を示す。

【００１６】

図１に戻り、キャッシュデータコントローラ６は、ＳＣＵ４からのキャッシュデータのリード／ライトリクエストを受け付けて、キャッシュデータ部７に対するキャッシュデータの読み書きを行う。パス１２はＳＣＵ４からのアクセスリクエストをキャッシュデータコントローラ６に送出するためのパスであり、パス１３はＳＣＵ４とキャッシュデータコントローラ６との間でデータをやりとりするためのパスである。パス３０はキャッシュデータコントローラ６がキャッシュデータ部７をリード／ライト制御するための信号線であり、パス３１はキャッシュデータコントローラ６とキャッシュデータ部７との間でデータをやり取りするためのパスである。本実施形態では、４ウエイセットアソシアティブキャッシュであるため、パス３０、３１の信号数が大きく、ＳＣＵ４から直接キャッシュデータ部７にピンをはることが物理的に不可能なため、データキャッシュコントローラ６をＳＣＵ４とは別チップとして構成しているものとする。これにより、パス１２は、１３はそれぞれパス３０、３１に比較して少ない信号数としている。

【００１７】

ＳＣＵ４内のメモリアクセスリクエストキュー１７はＣＰＵ（０）１、ＣＰＵ（１）２からのメモリアクセスリクエストをバッファリングして、コヒーレントコントローラ２０がビジーでない場合にメモリアクセスリクエストをコヒーレントコントローラ２０に送るためのキューである。データバッファ１８はＣＰＵ（０）１、ＣＰＵ（１）２からのライトデータを一時的に格納するバッファであり、データバッファ１９はＣＰＵ（０）１、ＣＰＵ（１）２に返すリードリプライデータを一時的に格納するためのバッファである。コヒーレントコントローラ２０は、ＣＰＵ（０）１、ＣＰＵ（１）２からのメモリアクセスリクエストがキャッシュヒットするかどうかを判定して、キャッシュデータコントローラ６とメモリアクセスコントローラ２１にアクセスリクエストを発行する。メモリアクセスコントローラ２１は、コヒーレントコントローラ２０からのアクセスリクエストに従って、メインメモリ８のアクセス制御を行う。

【００１８】

ここで、コヒーレントコントローラ２０では、図２に示したようなＣＰＵ（０）１およびＣＰＵ（１）２からのリクエストアドレス１００を分解し、キャッシュエントリ番号１０２を使って図３に示すキャッシュタグ部５の該当するエントリの各ウエイからキャッシュタグ２１１を読み出し、これらとリクエストアドレス１００のキャッシュタグ１０１と比較することによってキャッシュヒットチェックを行う。このキャッシュヒットチエック自体は従来と基本的に同様である。

【００１９】

次に、図１のコンピュータシステムにおいて一実施の形態の動作を説明する。

ＣＰＵ（０）１、ＣＰＵ（１）２では、命令実行に必要なデータが内臓キャッシュにない場合、メモリアクセスリクエストをバス３を介してＳＣＵ４に発行する。ＳＣＵ４では、バス１６を介してメモリアクセスリクエストキュー１７に該メモリアクセスリクエストを格納する。ライトリクエストの場合にはＣＰＵ（０）１、ＣＰＵ（１）２からデータも送出されるため、ＳＣＵ４ではライトデータをバス１６を介してデータバッファ１８に格納する。コヒーレントコントローラ２０がビジーでない場合、メモリアクセスリクエストキュー１７からコヒーレン

トコントローラ２０にメモリアクセスリクエストを送出する。

【００２０】

コヒーレントコントローラ２０では、キャッシュタグ部５を参照して、受け取ったメモリアクセスリクエストがキャッシュにヒットするかどうかを判定するが、リードリクエストの場合には、キャッシュヒット判定結果を待ってから、キャッシュデータコントローラ６を介して、データキャッシュ７からデータを読み出すと、アクセスレイテンシが大きくなってしまう。そこで、コヒーレントコントローラ２０は、キャッシュタグ部５によるキャッシュヒット判定を行う前に、キャッシュデータコントローラ６に対して投棄的なリードを行うリクエストを発行し、ヒットした場合に読み出す必要のあるデータをあらかじめキャッシュデータ部７からキャッシュデータコントローラ６まで読み出しておいて、ヒットした場合にはこの先読みしたデータを使用する。

【００２１】

図５は、コヒーレントコントローラ２０の一実施の形態の処理フローである。以下に、上記コヒーレントコントローラ２０の詳細な動作を図５を用いて説明する。

【００２２】

コヒーレントコントローラ２０では、ＣＰＵ（０）あるいはＣＰＵ（１）２からのメモリアクセスリクエストを受け付けると（ステップ３００）、そのリクエストがリードかどうかを判定する（ステップ３０１）。もしリードリクエストであれば、パス２５，１２を介してキャッシュデータコントローラ６に投棄リードリクエストを発行し（ステップ３０２）、同時にキャッシュタグ部５にパス２３，１０を介して当該リードリクエストのキャッシュエントリ番号を送り、キャッシュタグ部５からパス１１，２４を介して当該エントリの４ウエイ分のキャッシュタグの読み出しを行う（ステップ３０３）。このキャッシュタグ部５から読み出したキャッシュタグがリードリクエストのキャッシュタグとヒットするかどうかを判定し（ステップ３０４）、ヒットした場合には、パス２５，１２を介してキャッシュデータコントローラ６にリードリクエストを発行する（ステップ３０５）。このときのリードリクエストには、キャッシュエントリ番号とともにヒッ

トしたウエイ番号が含まれる。キャッシュミスした場合には、メモリアクセスコントローラ２１にリードリクエストを発行し（ステップ３０６）、パス２３，１０を介してキャッシュタグ部５の該当エントリの所望ウエイに該メモリアクセスリクエストのキャッシュタグを新しく登録する（ステップ３０７）。メモリアクセスコントローラ２１はパス２７，１４を介してメインメモリ８をアクセスし、パス１５，２８にデータを読み出す。このメインメモリ８からのリプライデータが帰ってきたら、コヒーレントコントローラ２０では、パス２５，１２を介し、キャッシュデータコントローラ６に対して、このリプライデータをキャッシュデータ部７に登録するライトリクエストを発行し、パス２２、パス２６，１３を介し、当該リプライデータをライトデータとしてキャッシュデータコントローラ６に送出する（ステップ１０８）。同時に、リプライデータをＣＰＵに送るために、パス２２よりデータバッファ１９に当該リプライデータを格納する（ステップ３０９）。

【００２３】

ＣＰＵ（０）１あるいはＣＰＵ（１）２から受け取ったリクエストがライトリクエストであった場合、コヒーレントコントローラ２０では、リードリクエストの場合と同様にしてキャッシュタグ部５から当該エントリの４ウエイ分のキャッシュタグを読み出し（ステップ３１０）、キャッシュヒットしているかどうかを判定する（ステップ３１１）。ヒットしていた場合、パス２５，１２を介してキャッシュデータコントローラ６にライトリクエストを発行し（ステップ３１２）、同時にデータバッファ１８からパス２２、パス２６，１３を介してライトデータをキャッシュデータコントローラ６に送出する（ステップ３１３）。このときのライトリクエストには、キャッシュエントリ番号とともにヒットしたウエイ番号が含まれる。ミスしていた場合には、メモリアクセスコントローラ２１にライトリクエストを発行し（ステップ３１４）、同時にデータバッファ１８からパス２２、パス２８，１５を介してメインメモリ８にライトデータを送出する（ステップ３１５）。メモリアクセスコントローラ２１では、パス２７，１４を介してメインメモリ８をアクセスし、メインメモリ８にデータを書き込む。

【００２４】

このように、コヒーレントコントローラ２０では、特にリードリクエストを受け付けた場合に、キャッシュタグ部５によるキャッシュヒットチェックを行う前にキャッシュデータコントローラ６に投棄的リードリクエストを発行する機能を持つ。ライトリクエストの場合には基本的に従来と同様である。

【００２５】

次に、キャッシュデータコントローラ６の構成および動作を説明する。図１において、キャッシュデータコントローラ６はパス１２を介してコヒーレントコントローラ２０からの投棄的リードリクエスト、リードリクエスト、ライトリクエストに従って、キャッシュデータ部７とデータのやりとりを行う。

【００２６】

図６は、キャッシュデータコントローラ６の詳細なブロック図である。キャッシュデータコントローラ６は、リクエストコントローラ４００、投棄リードリクエストバッファ４０１、アドレスコンパレータ部４０２、投棄リードデータバッファ４０３～４０６、バス４０７，４０８～４１１、セレクタ４１２，４１３～４１６、パス４１７～４２８からなる。

【００２７】

リクエストコントローラ４００では、パス１２を介してコヒーレントコントローラ２０から受け取ったリクエストをデコードし、該受け付けたリクエストの種類によって当該キャッシュデータコントローラ６で行う処理を決定し、各部を制御する。投棄リードリクエストバッファ４０１は、コヒーレントコントローラ２０から受け取った投棄リードリクエストを保持するバッファである。投棄リードデータバッファ４０３～４０６は投棄リードリクエストに従ってキャッシュデータ部７から読み出したデータを保持するバッファである。図４に示した通り、本実施形態のキャッシュデータ部７は４ウエイセットアソシアティブであり、第０ウエイのデータは投棄リードデータバッファ４０３に、第１ウエイのデータは４０４に、第２ウエイのデータは４０５に、第３ウエイのデータは４０６に格納する。アドレスコンパレータ部４０２は、コヒーレントコントローラ２０から受け取った投棄リードリクエスト、リードリクエスト・ライトリクエストが、投棄リ

ードリクエストバッファ４０１に格納されたリクエストと同一キャッシュエントリであるかどうかをチェックする。

【００２８】

図７および図８に投棄リードリクエストバッファ４０１と投棄リードデータバッファ４０３～４０６の構成例を示す。図７に示すように、投棄リードリクエストバッファ４０１は複数のエントリからなり、各エントリはバリッドビット（Ｖ）５００、キャッシュエントリ番号５０１からなる。バリッドビット５００はそのエントリが有効・無効であることを示すビットであり、キャッシュエントリ番号５０１は当該エントリに格納されている投棄リードリクエストが対象とするキャッシュエントリ番号である。投棄リードデータバッファ４０３～４０６も、図８に示すように複数のエントリからなり、各エントリには、投棄リードリクエストによりキャッシュデータ部７から投棄的に読み出したキャッシュデータ（３２Ｂ）６００が格納されている。

【００２９】

投棄リードリクエストバッファ４０１のエントリと投棄リードデータバッファ４０３～４０６のエントリは一対一に対応する。例えば、投棄リードリクエストバッファ４０１の０エントリに、ある投棄リードリクエストのキャッシュエントリ番号が格納されているとすると、当該投棄リードリクエストによりキャッシュデータ部７から投棄的に読み出された４ウエイ分キャッシュデータは、投棄リードデータバッファ４０３～４０６の０エントリに格納される。なお、これら投棄リードリクエストバッファ４０１と投棄リードデータバッファ４０３～４０６のエントリ数（ｍ）は任意でよい。また、これらバッファ４０１，４０３～４０６は一体的に構成してもよい。

【００３０】

図９はリクエストコントローラ４００の一実施形態の処理フローである。以下に、図９により、リクエストコントローラ４００を中心にキャッシュデータコントローラ６の詳細な動作を説明する。

【００３１】

リクエストコントローラ４００では、コヒーレントコントローラ２０からパス

１２、４１７を通してリクエストを受け取ると（ステップ７００）、まず、投棄リードリクエストかどうかをチェックする（ステップ７０１）。投棄リードリクエストの場合、投棄リードリクエストバッファ４０１に同一キャッシュエントリへのリクエストが格納されているかチェックする（ステップ７０２）。具体的には、リクエストコントローラ４００は、投棄リードリクエストのキャッシュエントリ番号をパス４１９に出力するとともに、投棄リードリクエストバッファ４０１の各エントリのキャッシュエントリ番号を読み出し、両者をアドレスコンパレータ部４０２で比較し、比較結果をパス４２０で受け取ることで、投棄リードリクエストバッファ４０１に投棄リードリクエストと同一キャッシュエントリが格納されているかチエックする。ここで、もし格納されていれば、新しく受け取った投棄リードリクエストを無視する。もしも投棄リードリクエストバッファ４０１に同一キャッシュエントリへのリクエストが格納されていない場合、投棄リードリクエストバッファ４０１がフルかどうかをチェックする（ステップ７０３）。もしフルでなければ、パス２８を通して、投棄リードリクエストバッファ４０１内の空きエントリに新しい投棄リードリクエストを登録し（ステップ７０５）、もしフルであれば、投棄リードリクエストバッファ４０１内の最も古いエントリを無効化した後（ステップ７０４）、新しいリクエストを登録する。なお、この種の無効化のアルゴリズムはＬＲＵ（Least Recentry Used）法として良く知られており、詳しい説明は省略する。この登録した投棄リードリクエストを、パス２１８、３０を介してキャッシュデータ部７にリードリクエストとして転送し、キャッシュデータ部７の該当キャッシュエントリから４ウエイ分のキャッシュデータを読み出し（ステップ７０６）、パス３１、バス４０８～４１１、パス４２３～４２６を介し、投棄リードデータバッファ４０３～４０６内の、投棄リードリクエストバッファ４０１に投棄リードリクエストを登録したエントリと対応するエントリに、新しく該キャッシュデータを格納する（ステップ７０７）。この結果、投棄リードリクエストバッファ４０１がフルの場合には、該投棄リードリクエストバッファ４０１の無効化したエントリに対応するところの、投棄リードデータバッファ４０３～４０６の該当エントリに新しいキャッシュデータが上書きされて格納されることになる。

【００３２】

コヒーレントコントローラ２０から受け取ったリクエストが投棄リードリクエストでなくリードリクエストであった場合には（ステップ７０８）、まず、投棄リードリクエストバッファ４０１にリードリクエストと同一キャッシュエントリのアドレス（キャッシュエントリ番号）が格納されているかチェックする（ステップ７０８）。チエックの仕方は投棄リードリクエストの場合と同様である。もしあれば、投棄リードデータバッファ４０３～４０６の該当エントリからデータを読み出し、セレクタ４１３～４１６、セレクタ４１２、バス４０７を介し、リプライデータとしてパス１３にデータを送りだす（ステップ７１０）。即ち、リクエストコントローラ４００は、パス４２２に投棄リードリクエストバッファ側の選択信号を出力し、パス４２１にリードリクエストに含まれるヒットウエイ番号を選択信号として出力する。これにより、投棄リードデータバッファ４０３～４０６の該当エントリから読み出された４ウエイ分のデータが、まずセレクタ４１３～４１６で選択され、次に、該４ウエイ中のヒットウエイ番号に対応するデータがセレクタ４１２で選択され、リプライデータとしてバス４０７を介してパス１３に送出される。その後、投棄リードリクエストバッファ４０１の該当エントリを無効化する（ステップ７１１）。

【００３３】

もし投棄リードリクエストバッファ４０１にリードリクエストと同一キャッシュエントリのアドレスがない場合、リクエストコントローラ４００では、パス２１８、３０を介してキャッシュデータ部７に当該リードリクエストを転送し、キャッシュデータ部７の該当キャッシュエントリから読み出された４ウエイ分のキャッシュデータをバス４０８～４１１を介してセレクタ４１３～４１６で選択し、そのうちのヒットウエイ番号に対応するデータをセレクタ４１２で選択して、リプライデータとしてバス２０７からパス１３に送り出す（ステップ７１２）。なお、このケースは、投棄リードリクエストによりキャッシュデータ部７から投棄リードデータバッファ４０３～４０６に先き読みしたデータが、その後の対応するリードリクエストの前に次に説明するようなライトリクエスト（先行ライトリクエスト）により無効化されていた場合に発生する。

【００３４】

コヒーレントコントローラ２０からパス１２，２１７を介して受け取ったリクエストが投棄リードリクエストでもリードリクエストでもない場合、すなわちライトリクエストの場合も、投棄リードリクエストバッファ４０１に同一キャッシュエントリへのアドレスが格納されているかチェックする（ステップ７１３）。もしあれば、投棄リードリクエストバッファ４０１の該当エントリを無効化する（ステップ７１４）。次に、パス２１８、３０を介してキャッシュデータ部７にライトリクエストを送出し、同時にコヒーレントコントローラ２０からパス１３を介して受け取ったキャッシュデータをパス２０７、パス４２７、パス２０８～２１１を介してパス３１に送出し、キャッシュデータ部７の指定されたエントリの指定されたウエイ番号に該データを書き込む（ステップ７１５）。

【００３５】

図９中、ステップ７１４において、コヒーレントコントローラ２０から受け取ったライトリクエストと同一エントリへのリクエストが投棄リードリクエストバッファ４０１内にある場合に、該当エントリを無効化するのは、ライトによってキャッシュデータ部７内のデータが書き換えられてしまい、投棄リードデータバッファ４０３～４０６内のデータと不一致となってしまうためである。このステップ７１４の無効化処理により、同一キャッシュエントリに対する後続のリードレクエストでは、ステップ７１２において、キャッシュデータ部７から書き換えられた新しいデータが読み出されることになる。

【００３６】

本実施の形態では、図９に示すように、キャッシュデータコントローラ６は、コヒーレントコントローラ２０から受け取ったリードリクエストが、投棄リードリクエストバッファ４０１内のリクエストと同一エントリへのリクエストであった場合、キャッシュデータ部７からではなく、投棄リードデータバッファ４０３～４０６に先読みして格納されているデータを選択して返すことにより、キャッシュデータ部７のアクセスレイテンシを削減することができる。したがって、図５に示すように、コヒーレントコントローラ２０では、キャッシュヒットチェックを行っている間に、投棄リードを発行しておけば、キャッシュヒットチェック

時間分のサイクルをメモリアクセスレイテンシから削減することが可能となる。

【００３７】

図１０および図１１は本発明の他の実施形態におけるコヒーレントコントローラ２０とキャッシュデータコントローラ６内のリクエストコントローラ４００の処理フローを示したものである。

【００３８】

図１０はコヒーレントコントローラ２０の処理フローである。図１０が先の図５と異なる点は、ステップ８００が追加されたことである。ステップ８００は、ステップ３０２でキャッシュデータコントローラ６に発行した投棄リードリクエストがキャッシュミスだった場合、キャッシュデータコントロール６に対して、当該投棄リードリクエストで先読みした投棄リードデータを無効化するリクエスト（投棄リードデータ破棄リクエスト）を発行するものである。これにより、キャッシュデータコントロール６では、投棄リードデータバッファ４０２内の使用されない投棄リードデータを無効化することができるため、投棄リードリクエストバッファ４０１及び投棄リードデータバッファ４０２の有効利用が可能になる。

【００３９】

図１１はキャッシュデータコントローラ６内のリクエストコントローラ４００の処理フローである。図１１が先の図９と異なる点は、ステップ９００と９０１が追加されたことである。このステップ９００と９０１がコヒーレントコントローラ２０から投棄リードデータ破棄リクエストを受け付けた場合の処理フローである。即ち、リクエストコントローラ４００では、コヒーレントコントローラ２０から投棄リードデータ破棄リクエストを受け取ったならば（ステップ９００）、投棄リードリクエストバッファ４０１内の当該投棄リードデータ破棄リクエストに対応する投棄リードリクエストのキャッシュエントリ番号が登録されているエントリを無効化する（ステップ９０１）。これにより、投棄リードリクエストバッファ４０１及び投棄リードデータバッファ４０２の有効利用が可能になり、該バッファ４０１，４０２のエントリ数をある程度余裕をもって構成すると、フル状態を解消し、ステップ７０３、７０４のフル制御自体を不要とすることも可

能になる。

【００４０】

本実施の形態においても、先の実施の形態１と同様に、キャッシュデータコントローラ６では、コヒーレントコントローラ２０から受け取ったリードリクエストが、投棄リードリクエストバッファ４０１内の投棄リードリクエストと同一エントリへのリクエストであった場合、キャッシュデータ部７からではなく、投棄リードデータバッファ４０３～４０６のいずれかからデータを読み出すことにより、キャッシュデータ部７のアクセスレイテンシを削減することができる。したがって、コヒーレントコントローラ２０においてキャッシュヒットチェックを行っている間に、投棄リードリクエストを発行しておけばキャッシュヒットチェック時間分のサイクルをメモリアクセスレイテンシから削減することが可能となる。

【００４１】

以上、本発明の実施の形態では、キャッシュは４ウエイセットアソシアティブであるとしたが、一般にウエイ数は１以上いくつでもよく、また、本発明は、セットアソシアティブキャッシュに限らず、キャッシュタグ部分とキャッシュデータ部を別ＬＳＩ等で管理するキャッシュ方式を採用したコンピュータシステムに広く適当可能であることは云うまでもない。

【００４２】

【発明の効果】

以上説明したように、本発明によれば、大容量キャッシュ等を実現するため、キャッシュタグ部分とキャッシュデータ部分を別ＬＳＩ等で管理した場合のキャッシュデータの読み出し時間を短縮することが可能になる。

【図面の簡単な説明】

【図１】

本発明の一実施形態のコンピュータシステムを示すブロック図である。

【図２】

ＣＰＵからのアドレスとキャッシュタグ、キャッシュエントリ番号の関係を示す図である。

【図３】

キャッシュタグ部の構成例を示す図である。

【図４】

キャッシュデータ部の構成例を示す図である。

【図５】

本発明の第１の実施形態によるコヒーレントコントローラの処理フロー図である。

【図６】

キャッシュデータコントローラの詳細ブロック図である。

【図７】

キャッシュデータコントローラ内の投棄リードリクエストバッファの構成例を示す図である。

【図８】

キャッシュデータコントローラ内の投棄リードデータバッファの構成例を示す図である。

【図９】

本発明の第１の実施形態によるキャッシュデータコントローラ内のリクエストコントローラの処理フロー図である。

【図１０】

本発明の第２の実施形態によるコヒーレントコントローラの処理フロー図である。

【図１１】

本発明の第２の実施形態によるキャッシュデータコントローラ内のリクエストコントローラの処理フロー図である。

【図１２】

従来の４ウエイセットアソシアティブキャッシュの構成例を示す図である。

【符号の説明】

１，２　ＣＰＵ

５　　キャッシュタグ部

６　キャッシュデータコントローラ

７　キャッシュデータ部

８　メインメモリ

２０　コヒーレンスコントローラ

２１　メモリアクセスコントローラ

４００　リクエストコントローラ

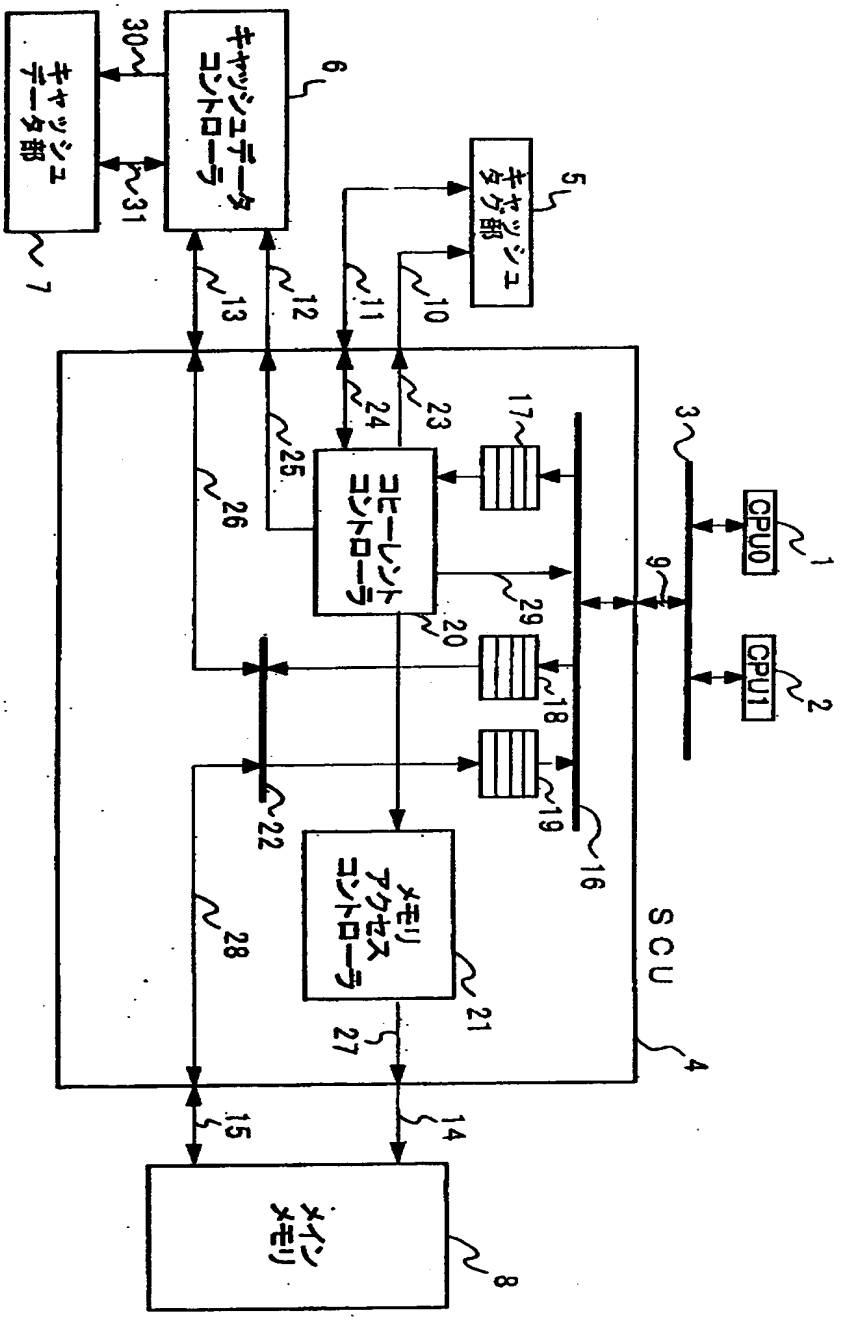
４０１　投棄リードリクエストバッファ

４０２　アドレスコンパレータ

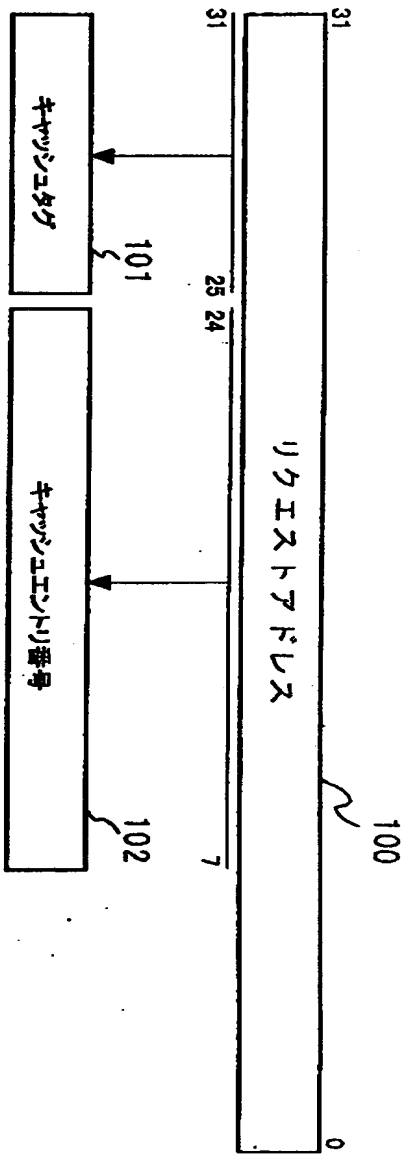
４０３～４０６　投棄リードデータバッファ

【書類名】　　図面

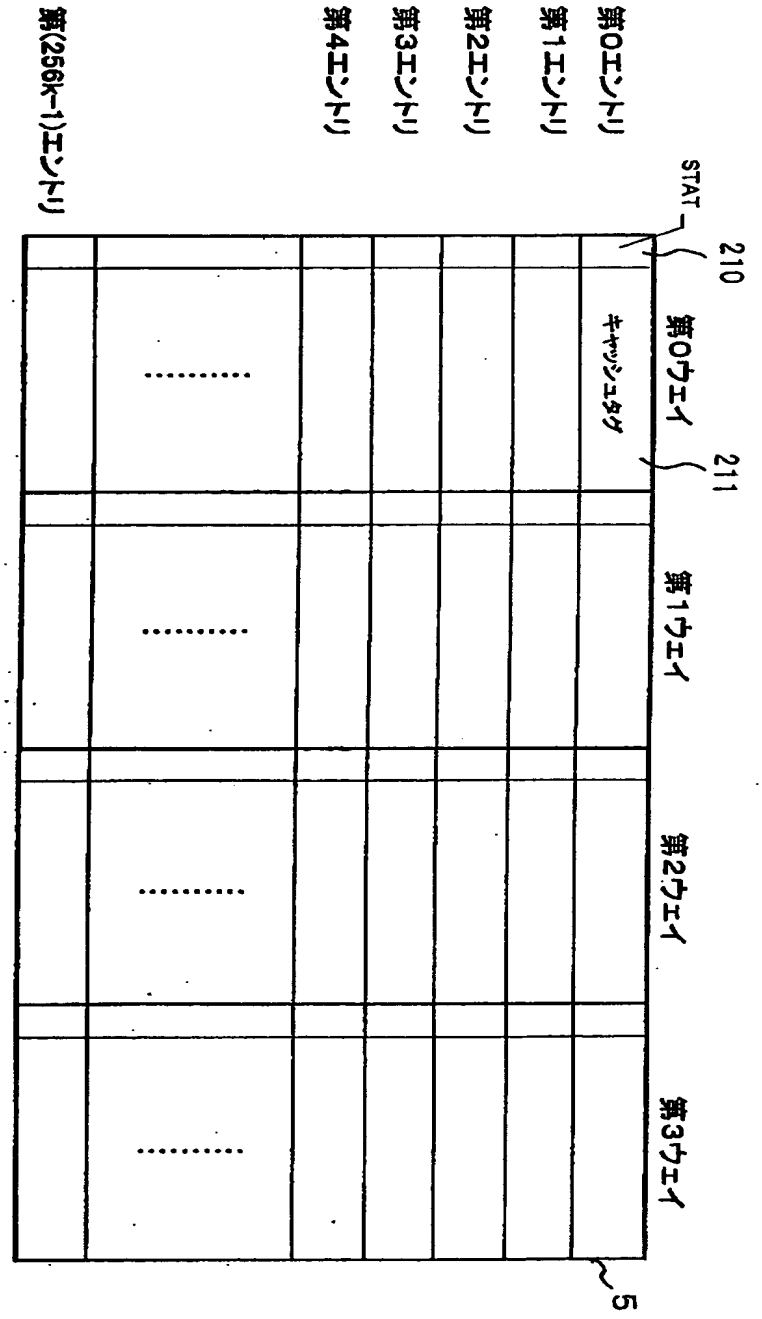
【図１】


キャッシュデータ部
キャッシュデータコントローラ
キャッシュタグ部
CPU0
CPU1
コヒーレントコントローラ
メモリアクセスコントローラ
SCU
メインメモリ
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

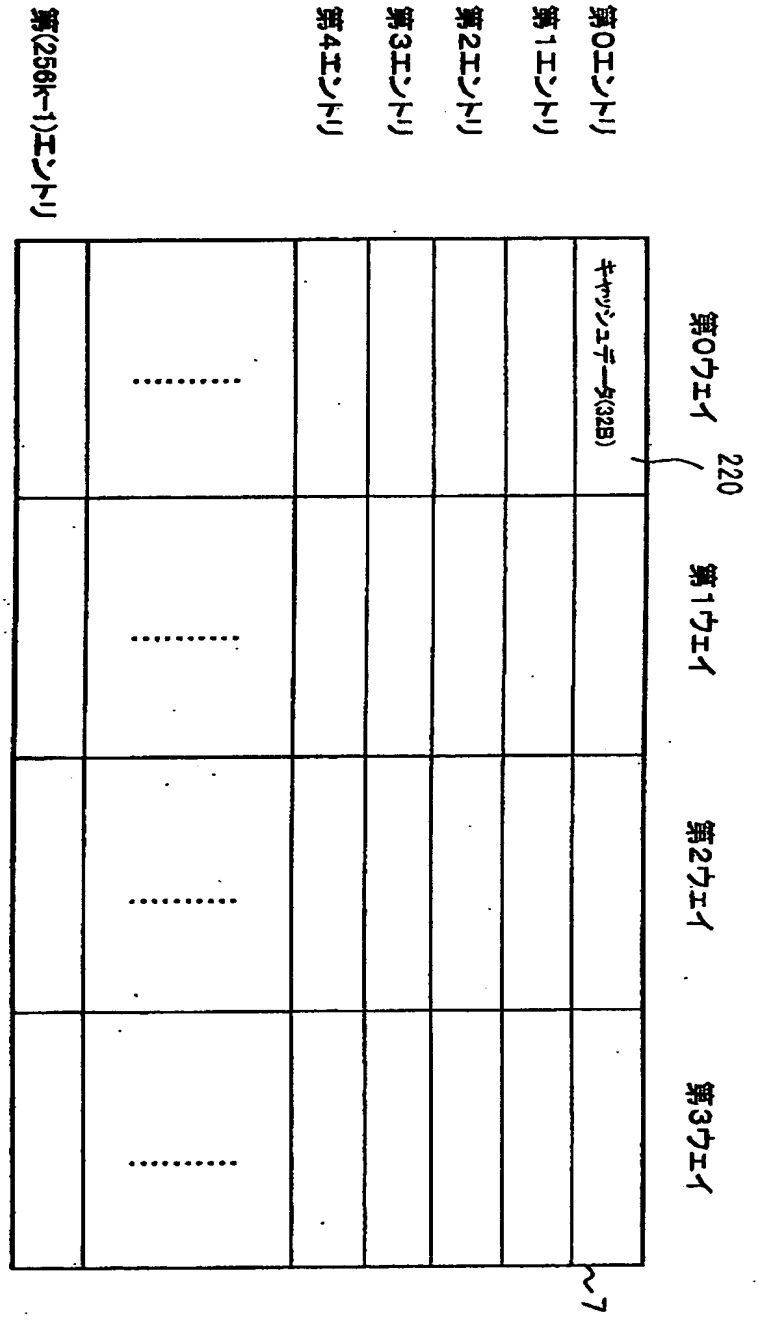
【図２】



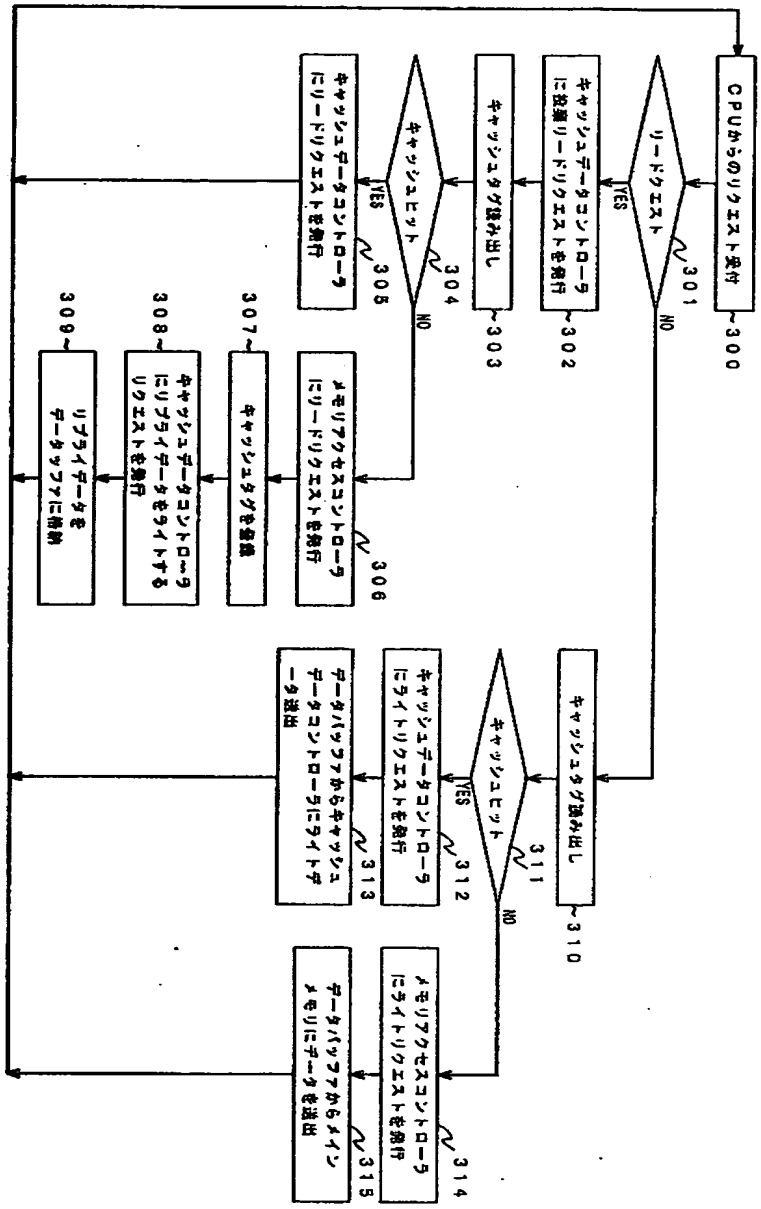
【図３】


第0エントリ
第1エントリ
第2エントリ
第3エントリ
第4エントリ
第(256k-1)エントリ
STAT
210
キャッシュタグ
第0ウェイ
211
第1ウェイ
第2ウェイ
第3ウェイ
5

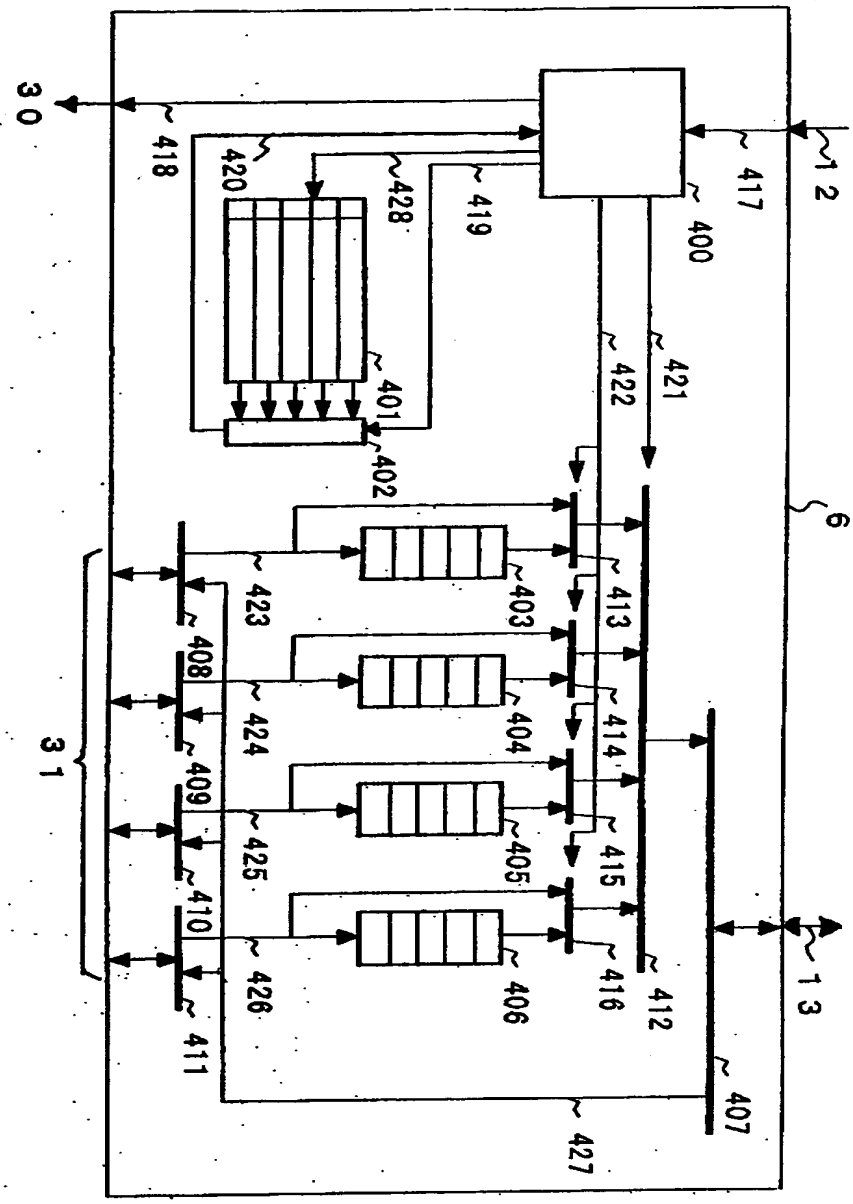
【図４】


第0エントリ
第1エントリ
第2エントリ
第3エントリ
第4エントリ
第(256k-1)エントリ
第0ウェイ
第1ウェイ
第2ウェイ
第3ウェイ
キャッシュデータ(32B)
220
7

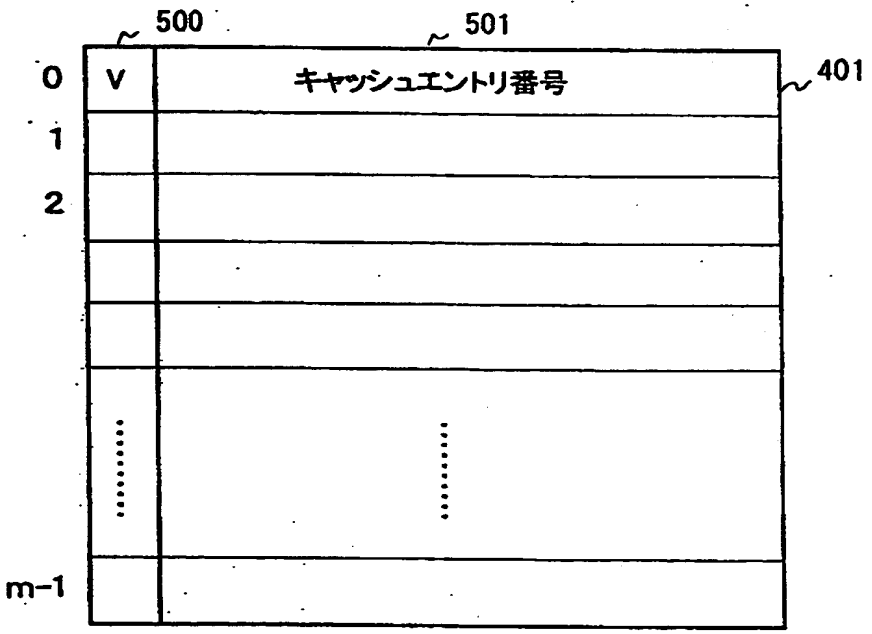
【図５】


CPUからのリクエスト受付
300
リードリクエスト
301
YES
NO
キャッシュデータコントローラに該当リードリクエストを発行
302
キャッシュタグ読み出し
303
キャッシュヒット
304
YES
NO
キャッシュデータコントローラにリードリクエストを発行
305
メモリアクセスコントローラにリードリクエストを発行
306
キャッシュタグを登録
307
キャッシュデータコントローラにリプライデータをライトするリクエストを発行
308
リプライデータをデータバッファに格納
309
キャッシュタグ読み出し
310
キャッシュヒット
311
YES
NO
キャッシュデータコントローラにライトリクエストを発行
312
データバッファからキャッシュデータコントローラにライトデータ送出
313
メモリアクセスコントローラにライトリクエストを発行
314
データバッファからメインメモリにデータを送出
315

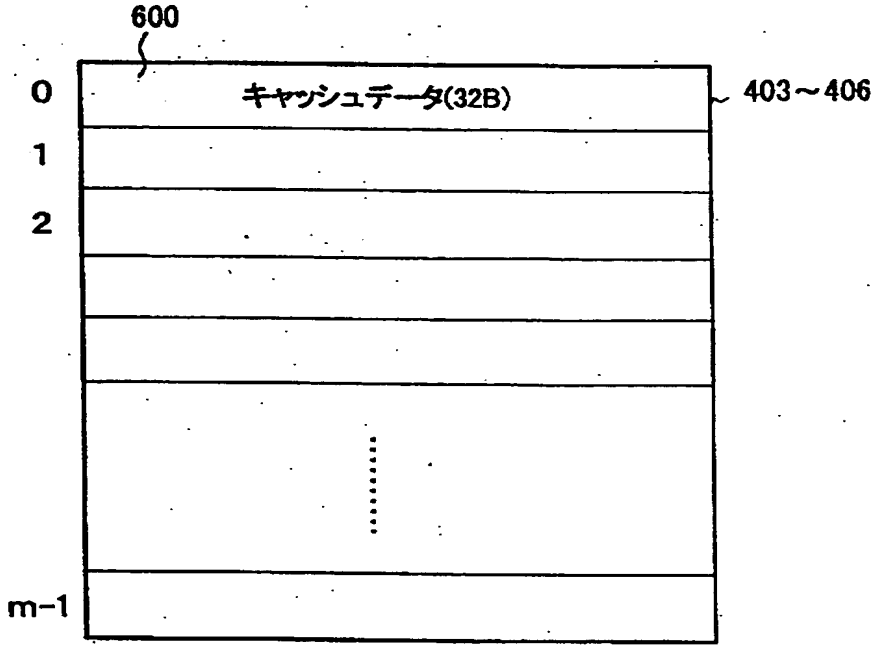
【図６】


30
418
420
428
419
417
12
400
422
421
401
402
6
423
403
413
408
424
404
414
31
409
425
405
415
410
426
406
416
412
411
13
407
427

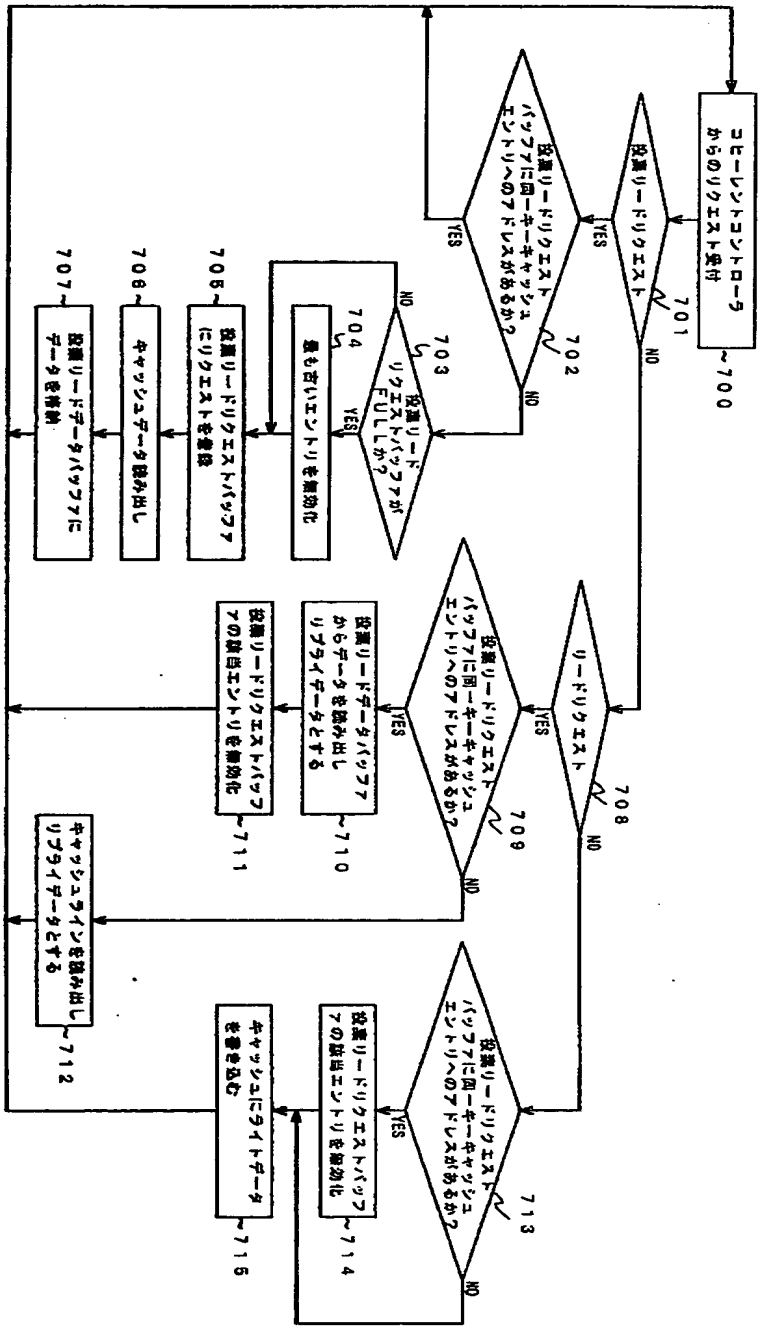
【図７】


500
501
0
V
キャッシュエントリ番号
401
1
2
m-1

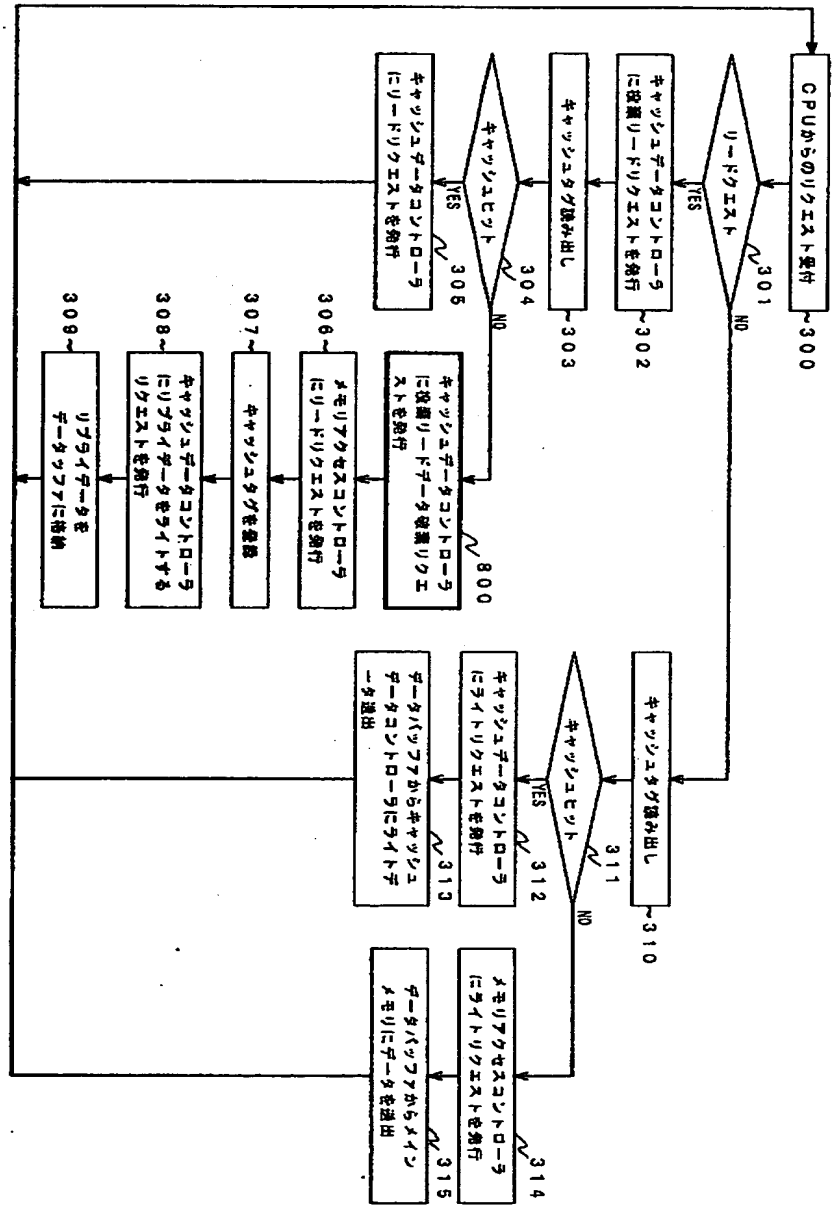
【図８】


600
0
キャッシュデータ(32B)
403～406
1
2
m-1

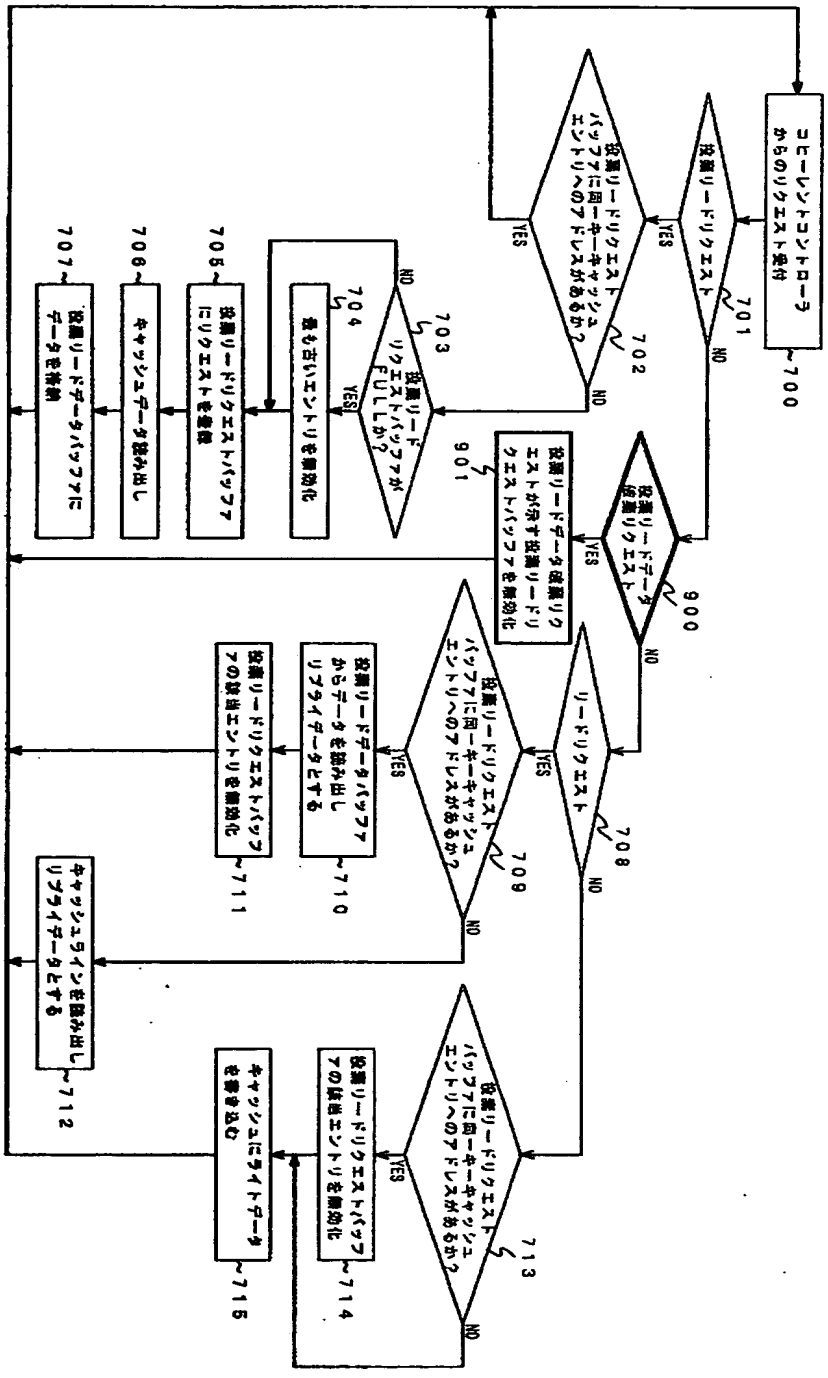
【図９】


コヒーレントコントローラからのリクエスト受付
700
投票リードリクエスト
701
YES
NO
投票リードリクエストバッファに同一キーキャッシュエントリへのアドレスがあるか？
702
YES
NO
投票リードリクエストバッファがFULLか？
703
YES
NO
最も古いエントリを無効化
704
投票リードリクエストバッファにリクエストを登録
705
キャッシュデータ読み出し
706
投票リードデータバッファにデータを格納
707
リードリクエスト
708
YES
NO
投票リードリクエストバッファに同一キーキャッシュエントリへのアドレスがあるか？
709
YES
NO
投票リードデータバッファからデータを読み出しリプライデータとする
710
投票リードリクエストバッファの該当エントリを無効化
711
キャッシュラインを読み出しリプライデータとする
712
投票リードリクエストバッファに同一キーキャッシュエントリへのアドレスがあるか？
713
YES
NO
投票リードリクエストバッファの該当エントリを無効化
714
キャッシュにライトデータを書き込む
715

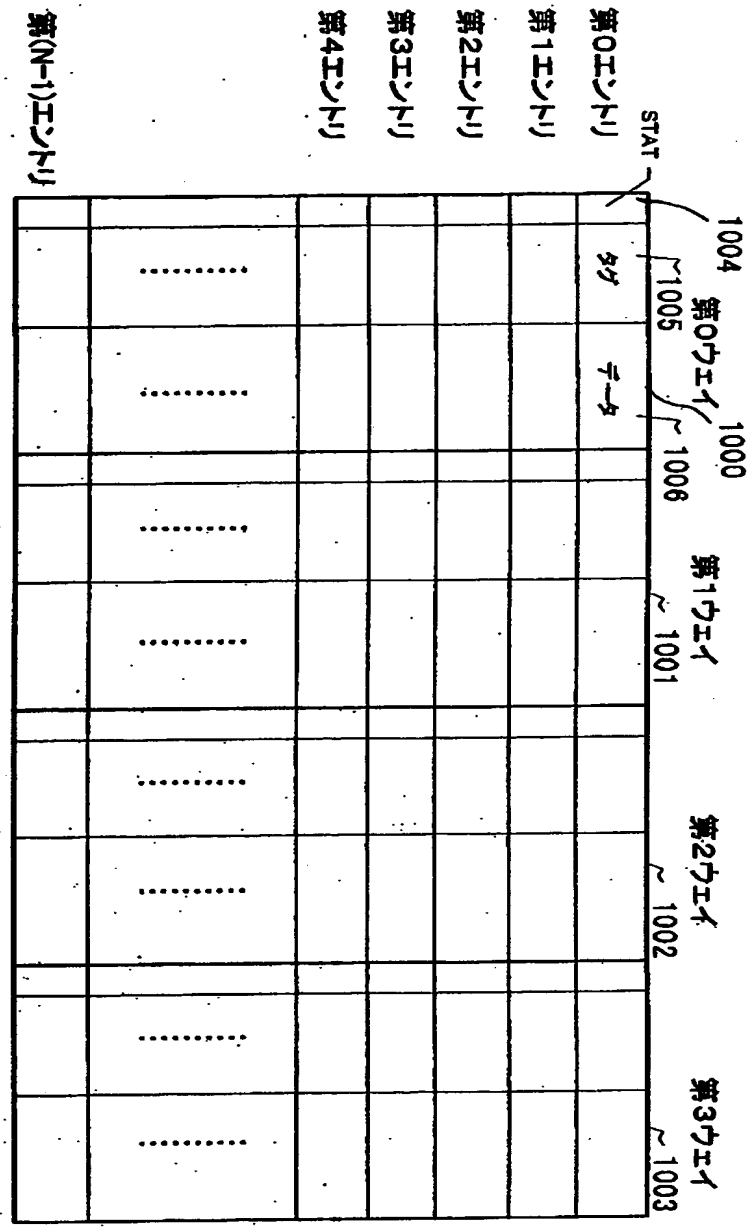
【図１０】


CPUからのリクエスト受付
300
リードクエスト
301
YES
NO
キャッシュデータコントローラに読出リードリクエストを発行
302
キャッシュタグ読み出し
303
キャッシュヒット
304
YES
NO
キャッシュデータコントローラにリードリクエストを発行
305
キャッシュデータコントローラに読出リードデータ収集リクエストを発行
800
メモリアクセスコントローラにリードリクエストを発行
306
キャッシュタグを登録
307
キャッシュデータコントローラにリプライデータをライトするリクエストを発行
308
リプライデータをデータバッファに格納
309
キャッシュタグ読み出し
310
キャッシュヒット
311
YES
NO
キャッシュデータコントローラにライトリクエストを発行
312
データバッファからキャッシュデータコントローラにライトデータ送出
313
メモリアクセスコントローラにライトリクエストを発行
314
データバッファからメインメモリにデータを送出
315

【図１１】


コヒーレントコントローラからのリクエスト受付
700
投票リードリクエスト
701
YES
NO
投票リードリクエストバッファに同一キーキャッシュエントリへのアドレスがあるか？
702
YES
NO
投票リードリクエストバッファがFULLか？
703
NO
YES
最も古いエントリを無効化
704
投票リードリクエストバッファにリクエストを登録
705
キャッシュデータ読み出し
706
投票リードデータバッファにデータを格納
707
投票リードデータ投票リクエスト
900
YES
NO
投票リードデータ投票リクエストが示す投票リードリクエストバッファを無効化
901
リードリクエスト
708
YES
NO
投票リードリクエストバッファに同一キーキャッシュエントリへのアドレスがあるか？
709
YES
NO
投票リードデータバッファからデータを読み出しリプライデータとする
710
投票リードリクエストバッファの該当エントリを無効化
711
キャッシュラインを読み出しリプライデータとする
712
投票リードリクエストバッファに同一キーキャッシュエントリへのアドレスがあるか？
713
YES
NO
投票リードリクエストバッファの該当エントリを無効化
714
キャッシュにライトデータを書き込む
715

【図１２】


STAT
第0エントリ
第1エントリ
第2エントリ
第3エントリ
第4エントリ
第(N-1)エントリ
1004
第0ウェイ
1000
タグ
データ
1005
1006
第1ウェイ
1001
第2ウェイ
1002
第3ウェイ
1003

【書類名】　　要約書

【要約】

【課題】　キャッシュ方式のコンピュータシステムにおいて、キャッシュタグ部とキャッシュデータ部が分割管理された場合のキャッシュデータの読み出しレイテンシを削減する。

【解決手段】　コヒーレントコントローラ２０は、メモリリード処理の場合、キャッシュタグ部５からキャッシュタグを読み出してキャッシュヒットチェックを行う前に、キャッシュデータコントローラ６に対して、キャッシュデータ部７からデータを先読み（投棄リード）しておく投棄リードリクエストを発行し、キャッシュデータコントローラ６では、キャッシュデータ部７から投棄リードしたデータを保持しておき、キャッシュヒットした場合にコヒーレントコントローラ２０から発行されるリードリクエストを受け取った時点で、投棄リードしておいたデータをリプライデータとして返送する。

【選択図】　　図１

出　願　人　履　歴　情　報

識別番号　　〔０００００５１０８〕

１．変更年月日　　１９９０年　８月３１日
［変更理由］　新規登録
住　所　東京都千代田区神田駿河台４丁目６番地
氏　名　株式会社日立製作所